CPSC 468 Midterm Review

Chapter 1

0.1 (Turing Machine)**.** A k-tape Turing Machine M is described by a tuple (Γ, Q, δ). Assume $k \geq 2$, with 1 read-only input tape and $k-1$ work tapes. The last work tape is assumed to be the output tape.

- Γ the finite alphabet of symbols that M may have on its tapes. Assume that Γ contains at least $\{0, 1, \square, \triangleright\}$.
- Q a finite set of possible states M's state register may be in. Assume that Q contains a q_{start} and q_{halt}.
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{L, S, R\}^k$ a transition function for M that takes in the current state and each head's read, and outputs the next state, with $k-1$ writes on all the work tapes, and movements for all k tapes.

0.2 (Computing a Function)**.** Let $f : \{0,1\}^* \to \{0,1\}^*$ and $T : \mathbb{N} \to \mathbb{N}$, with M a TM. We say that M computes f if for every $x \in \{0,1\}^*$, if M is initialized to the start configuration on input x, then it halts with $f(x)$ on the output tape. We say M computes f in $T(n)$-time if its computation on every x requires at most $T(|x|)$ steps.

0.3 (Time Constructible)**.** A function $T : \mathbb{N} \to \mathbb{N}$ is time constructible if $T(n) \geq n$ and there is a TM M that computes the function $x \mapsto \llcorner T(|x|) \lrcorner$ in time $T(n)$. $T(n) \geq n$ is to allow the algorithm to read its input. Some time constructible functions are n, $n \log n$, n^2 and 2^n.

0.4. For every $f : \{0,1\}^* \to \{0,1\}$ and time constructible $T : \mathbb{N} \to \mathbb{N}$, if f is computable in time $T(n)$ by some TM M using alphabet Γ, then it is able to compute the same function using $\{0, 1, \square, \triangleright\}$ in $(c \log_2 |\Gamma|) \cdot T(n)$. This is because we may express each symbol of Γ using $\log |\Gamma|$ binary bits, with some constant c overhead.

0.5. A k-tape TM can have its $k-1$ work tapes simulated by a single tape by interleaving the k tapes together.

0.6 (Oblivious Turing Machine)**.** An oblivious TM's head movement depends on the length of the input, not the contents of the input. Every TM can be simulated by an oblivious TM.

0.7 (Turing Machine Representation)**.** Every binary string $x \in \{0,1\}^*$ represents some TM, and every TM is represented by infinite such strings (think: comments in a language). The machine represented by x is denoted M_x.

0.8 (Universal Turing Machine)**.** There exists a TM $\mathcal{U}$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}(x, a) = M_\alpha(x)$, where M_α denotes the TM represented by α. Moreover, if M_α halts on input x within T steps, then $\mathcal{U}_\alpha(x)$ halts within $CT \log T$ steps, where C is a number independent of $|x|$, and depends only on M_α's alphabet size, number of tapes, and number of states. The cost of simulating any machine M_α has a logarithmic overhead, due to the alphabet size difference between M_α and $\mathcal{U}$. As $\mathcal{U}$ has a single tape, we do the trick over interleaving M_α's work tapes together.

0.9 (Uncomputable Function)**.** Define U as follows: for every $\alpha \in \{0,1\}^*$, if the machine defined by α accepts itself, such that $M_\alpha(\alpha) = 1$, then $U(\alpha) = 0$. In other words $U(\alpha) = 1 - M_\alpha(\alpha)$. There is no such TM that can compute U, because U will always negate it.

0.10 (Halting Problem)**.** A TM H such that $H(\alpha, x) = 1$ if $M_\alpha(x)$ halts, and yields 0 otherwise, does not exist. We can construct a wrapper TM W that invokes H on itself, and performs the opposite. Diagonalization motherfuckers!

0.11 (DTIME)**.** Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in DTIME$(T(n))$ iff there is a deterministic TM that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L. This class contains **decision** problems.

0.12 (The Class P)**.** $\mathrm{P} = \bigcup_{c \geq 1} \mathrm{DTIME}(n^c)$

0.13 (Church-Turing Thesis)**.** Every physically realizable computation device can be simulated by a TM.

0.14 (Bounds)**.** The asymptoptic operators $\{o, O, \Theta, \Omega, \omega\}$ can be thought of as $\{<, \leq, =, \geq, >\}$.

Chapter 2

0.15 (Non-Deterministic Turing Machine)**.** A non-deterministic TM is endowed with two transition functions δ_0 and δ_1 along with a special accept state q_{accept}. The NDTM may use either transition function per time step. For every input x, we say that $M(x) = 1$ if there **exists** some sequence of transition function choises that would cause M to reach q_{accept}. Otherwise, if every sequence of non-deterministic choices causes M to halt on x without reaching q_{accept}, then we say that $M(x) = 0$. M runs in time $T(n)$ if for every input $x \in \{0,1\}^*$ and every sequence of non-deterministic choices, M reaches eitehr the halting state or q_{accept} within $T(|x|)$ steps.

0.16 (NTIME)**.** For every function $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$, we say that $L \in \text{NTIME}(T(n))$ if there is a constant $c > 0$ and a $c \cdot T(n)$-time NDTM M such that for every $x \in \{0,1\}^*$, we have $x \in L \iff M(x) = 1$.

0.17 (NP)**.** $\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)$

0.18. A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial time TM M (called the **verifier** for L) s.t. for every $x \in \{0,1\}^*$, we have $x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$. If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, we call u a **certificate** for x (with respect to L and M). NP is the class of languages for which we can tell if u is a solution to the problem $x \in L$ in polynomial time.

0.19 (Reductions, NP-hardness, and NP-completeness)**.** We say that a language $L \subseteq \{0,1\}^*$ is a polynomial time **Karp reducible** to a language $L' \subseteq \{0,1\}^*$ if there is a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, we have $x \in L \iff f(x) \in L'$. We say that L' is NP-hard if $L \leq_p L'$ for every $L \in \text{NP}$. We say that L' is NP-complete if L' is NP-hard and $L' \in \text{NP}$.

0.20 (Transitivity)**.** If $L \leq_p L'$, and $L' \leq_p L''$, then $L \leq_p L''$.

0.21. If L is NP-hard and $L \in \text{P}$, then $\text{P} = \text{NP}$.

0.22. If L is NP-Complete then $L \in \text{P} \iff \text{P} = \text{NP}$.

0.23 (TMSAT)**.** $\text{TMSAT} = \{(\alpha, x, 1^n, 1^t) : \exists u \in \{0,1\}^n \text{ s.t. } M_a(x,u) = 1 \text{ within } t \text{ steps}\}$

0.24 (EXP)**.** $\text{EXP} = \bigcup_{c>1} \text{DTIME}\left(2^{n^c}\right)$

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Examples